

MDT496 ESTUDIOS
DEPARTAMENTO DE OBRAS LITERARIAS
COLECCIÓN: DOCUMENTOS DE NORMALIZACIÓN
ISO - UNICODE - SI - IEC - ITU - AISC - ACI

Unicode

Una breve historia sobre codificación de caracteres

Marco Mendieta Parihuancollo

Primera Edición

© 2021 MDT496 ESTUDIOS

annlimma.atwebpages.com

youtube.com/mdt496

mdt496@gmail.com

Oruro, Bolivia

Unicode - Una breve historia sobre codificación de caracteres

Marco Mendieta Parihuancollo

Primera Edición (mdt496-ID: 036MP.AABD.009)



Esta monografía se distribuye bajo licencia Creative Commons: Atribución-NoComercial-SinDerivadas CC BY-NC-ND.

En particular, esta licencia permite usar, copiar, difundir bajo las siguientes condiciones: Citando la fuente, es decir al autor que lo escribió y el nombre del texto; No se permite uso comercial de la obra; No se permite modificar este material.

Dedicatoria

Esfuerzo sereno, triste pero siempre altivo, por encontrar en la energía misma de la vida el camino para aceptar la muerte.

Amado padre, hoy ausente, agradezco las muchas horas que estuviste a mi lado otorgando tu apoyo y protección. Al final, un silencio implacable, solo recuerdos que arañan el alma.

Este trabajo, producto de un gran esfuerzo, se lo dedico a mi amado padre **Marcos E. Mendieta C.**(†).

Contenido

Dedicatoria	III
Contenido	V
Lista de tablas	VII
Lista de figuras	IX
Glosario	XI
Introducción	XIII
Capítulo 1: Cosas que debe saber	1
1.1 Lo básico	1
1.2 Importancia	3
1.3 La codificación de caracteres	4
Capítulo 2: Sistemas de codificación	5
2.1 ASCII (ISO/IEC 646)	5
2.2 Latin-1 (ISO/IEC 8859-1)	7
2.3 Windows-1252	9
2.4 IBM 437	11
2.5 IBM 850 y 858	13
2.6 Otros sistemas	15
2.7 Comparativa	16
2.7.1 Latin-1 y Windows-1252	16
2.7.2 IBM 850 y 858	16
2.7.3 IBM 850/858 y 437	16
Capítulo 3: Unicode	19
3.1 La historia de Unicode	19
3.2 Character	19
3.3 Type	21
3.4 Charts	21
3.4.1 Basic Latin	28
3.4.2 Latin-1 Supplement	31
3.4.3 Latin Extended-A	34
3.4.4 Latin Extended-B	37
3.4.5 Greek and Coptic	41
3.4.6 Cyrillic	44
3.4.7 General Punctuation	48
3.4.8 Superscripts and Subscripts	50
3.4.9 Letterlike Symbols	52
3.4.10 Number Forms	54
3.4.11 Arrows	56
3.4.12 Mathematical Operators	59

3.4.13	Miscellaneous Technical	63
3.4.14	Enclosed Alphanumerics	67
3.4.15	Geometric Shapes	70
3.4.16	Miscellaneous Symbols	72
3.4.17	Miscellaneous Mathematical Symbols-A	76
3.4.18	Supplemental Arrows-A	78
3.4.19	Supplemental Arrows-B	80
3.4.20	Supplemental Mathematical Operators	83
3.4.21	Miscellaneous Symbols and Arrows	87
3.4.22	Latin Extended-C	89
3.4.23	Lisu	91
3.4.24	CJK Compatibility Forms	93
3.4.25	Small Form Variants	95
3.5	Transformation	96
3.5.1	UTF-8	96
3.5.2	UTF-16	96
Capítulo 4:	Aplicaciones	97
4.1	AutoCAD	97
4.2	Java	97
4.2.1	Character (java.lang)	97
4.2.2	String (java.lang) en Java 9, 11 y 14	97
4.2.3	La API Charset (java.nio)	97
4.2.4	Librería MDT496 (bo.mdt496.unicode)	98
4.3	HTML	98
4.4	L ^A T _E X	99
4.4.1	Codificación de entrada	99
4.4.2	Declaración de un carácter	100
4.4.3	Codificación de fundiciones	101
4.4.4	Fundición e idioma (el paquete babel)	103
4.4.5	Accessing all characters of a font	108
4.4.6	Font Commands	108
4.4.7	Font Char Position	119
Referencias		133

Lista de tablas

Tabla 2.1	ASCII (ISO/IEC 646)	6
Tabla 2.2	Latin-1 (ISO/IEC 8859-1)	8
Tabla 2.3	Windows-1252	10
Tabla 2.4	IBM 437	12
Tabla 2.5	IBM 850	14
Tabla 2.6	Lista de nombres de algunas tablas	15
Tabla 2.7	Diferencias IBM 850 y 437	17
Tabla 3.1	Unicode Type(30)	21
Tabla 3.2	Unicode Block(162) - Parte 1 de 5	23
Tabla 3.3	UnicodeBlock(162) - Parte 2 de 5	24
Tabla 3.4	UnicodeBlock(162) - Parte 3 de 5	25
Tabla 3.5	UnicodeBlock(162) - Parte 4 de 5	26
Tabla 3.6	UnicodeBlock(162) - Parte 5 de 5	27
Tabla 4.1	L ^A T _E X - fundiciones Computer Modern	101
Tabla 4.2	Unicode and L ^A T _E X Font Commands	110
Tabla 4.3	Font char position (0-127) OT1, T1, OML y OMS - Parte 1 de 4 . .	120
Tabla 4.4	Font char position (0-127) OT1, T1, OML y OMS - Parte 2 de 4 . .	121
Tabla 4.5	Font char position (0-127) OT1, T1, OML y OMS - Parte 3 de 4 . .	122
Tabla 4.6	Font char position (0-127) OT1, T1, OML y OMS - Parte 4 de 4 . .	123
Tabla 4.7	Font char position (0-255) T1, TS1, T2A, T2B, T2C y X2 - Parte 1 de 8	124
Tabla 4.8	Font char position (0-255) T1, TS1, T2A, T2B, T2C y X2 - Parte 2 de 8	125
Tabla 4.9	Font char position (0-255) T1, TS1, T2A, T2B, T2C y X2 - Parte 3 de 8	126
Tabla 4.10	Font char position (0-255) T1, TS1, T2A, T2B, T2C y X2 - Parte 4 de 8	127
Tabla 4.11	Font char position (0-255) T1, TS1, T2A, T2B, T2C y X2 - Parte 5 de 8	128
Tabla 4.12	Font char position (0-255) T1, TS1, T2A, T2B, T2C y X2 - Parte 6 de 8	129
Tabla 4.13	Font char position (0-255) T1, TS1, T2A, T2B, T2C y X2 - Parte 7 de 8	130
Tabla 4.14	Font char position (0-255) T1, TS1, T2A, T2B, T2C y X2 - Parte 8 de 8	131

Lista de figuras

Figura 1.1	Efectos de una declaración <code>charset</code> que no corresponde	3
Figura 1.2	Analizando el problema <code>charset</code> - <i>declaración</i>	4
Figura 3.1	Caracter Unicode	20
Figura 3.2	Una planilla Code Charts	22
Figura 4.1	Uso de Unicode en AutoCAD (Mapa de caracteres)	97
Figura 4.2	Caracteres Unicode para HTML	99
Figura 4.3	\LaTeX - <code>inputenc</code> - default (ASCII)	99
Figura 4.4	\LaTeX - <code>inputenc</code> - UTF-8	100
Figura 4.5	Redirección del punto de código (<code>DeclareUnicodeCharacter</code>) . . .	101
Figura 4.6	\LaTeX - <code>fontenc</code> - default	102
Figura 4.7	\LaTeX - <code>fontenc</code> - T1	102
Figura 4.8	\LaTeX - <code>fontenc</code> cyrillic	103
Figura 4.9	\LaTeX - <code>fontenc</code> y <code>lmodern</code>	103
Figura 4.10	\LaTeX - Fundición e idioma	104
Figura 4.11	\LaTeX - Ejemplo aplicado	105
Figura 4.12	\LaTeX - comandos <code>\mu</code> y <code>\times</code>	105
Figura 4.13	\LaTeX - <code>fontencoding</code>	108
Figura 4.14	\LaTeX - Font Commands	109
Figura 4.15	\LaTeX - Font Char Position	119

Glosario

A

American Standard Code for Information Interchange (ASCII) Forma de ordenar los caracteres que se pueden escribir utilizando un formato de 7 bits., pág. 5.

U

Unicode Unicode es un estandar creado en 1991 que pretende uncluir todos los caracteres de uso común. Un repositorio de más de 100 mil caracteres; Planillas de códigos para referencia visual (Code Charts); Metodologías de codificación; Codificaciones estándares; etc. The Unicode Standard / the Unicode Consortium; edited by the Unicode Consortium. Unicode and the Unicode Logo are registered trademarks of Unicode, Inc., in the United States and other countries., pág. 19.

Unicode Transformation Format 8-Bit (UTF-8) UTF-8 can be used to transmit text data through communications systems that assume that individual octets in the range of x00 to x7F have a definition according to ISO/IEC 4873, including a C0 set of control functions according to the 8-bit structure of ISO/IEC 2022. UTF-8 also avoids the use of octet values in this range that have special significance during the parsing of file name character strings in widely used file-handling systems., pág. 96.

Universal Multiple-Octet Coded Character Set (UCS) This 32-bit character encoding standard is for all practical purposes identical to Unicode standard. The layout of the Basic Multilinula Plane (plane 0 or BMP) is described in detail. This architecture is speciefies by [1]. Complementing [2], which describes plane 0 (BMP) of the UCS, the present standard details the layout of the supplementary planes;., pág. 96.

Introducción

Este libro describe el concepto de carácter, sistemas de codificación de caracteres y el estándar Unicode. El libro está dividido en cuatro capítulos.

En el capítulo 1 presentamos a los lectores las bases de la codificación de caracteres enfocado al desarrollo de software. El capítulo ilustra el concepto de carácter y sus atributos como *glifo* y *punto de código*; Justificamos su importancia y los efectos que produce el ignorarlo; Por último, presenta una reseña del proceso de intercambio de información, que sirve de prefacio al tema sistemas de codificación. Tras leer este capítulo, debería tener el conocimiento básico, que necesitará para entender el resto de este libro.

En el capítulo 2 se describe algunos sistemas de codificación. De cada una veremos atributos como, el nombre oficial (el que es designado por un organismo internacional) el canónico, es decir, por el que es más conocido y sus seudónimos (alias). La sección de comparativa no se trata de listar ventajas o desventajas de una tabla X, nos enfocaremos en mostrar coincidencia en *puntos de código* de la tabla X con la Y. Por ejemplo, Latin-1 y Windows-1252 coinciden en 96 de 128 caracteres que representa el 75 % cuyas posiciones 160 al 255.

El capítulo 3 trata del estándar Unicode. En su cuerpo nos explican un poco más sobre glifos o el uso de términos *code point* y *UTF-16 code unit*; Mostrar algunas de las plantillas que podrían ser útiles, por ejemplo: Los números 0-9 (U+0030 al U+0039) y las letras A-Z (U+0041 al U+005A), a-z (U+0061 al U+007A) pertenecen a la familia BASIC LATIN (U0000); La letra minúscula u con diéresis ü (U+00FC) está en el bloque LATIN-1 SUPPLEMENT (U0080); Algunas letras griegas como α (U+03B1), β (U+03B2), γ (U+03B3) están en la plantilla GREEK AND COPTIC (U0370). *Note el argumento entre paréntesis después del nombre de un grupo, esto no es casual, representa el identificador del primer carácter con el que inicia el bloque por lo tanto, es el identificador del bloque. Esto facilita la búsqueda porque le da el nombre al archivo en PDF, en este caso **U0370.pdf**.* Algunos caracteres cirílicos como Ж (U+0416) o л (U+043B) están en el bloque CYRILLIC (U0400).

El capítulo 4 veremos y aplicaremos todo lo aprendido en situaciones reales, por ejemplo,

```
\documentclass{article}
\usepackage[papersize={55mm,16mm}, margin=1mm]{geometry}
\usepackage{lmodern,xcolor}
\usepackage[T2A, T1]{fontenc}
\begin{document}
  \centering
  \char76\char111\
  \char113\char117\char101\
  \char102\char117\char101\
  \char118\char225\char108\char105\char100\char111\
  \char97\char121\char101\char114\
  \char115\char101\char114\char225\
  \char109\char97\char241\char97\char110\char97\
  \char111\char98\char115\char111\char108\char101\char116\char111\char46
```

```

\color{purple}\fontencoding{T2A}\fontfamily{cmtt}\selectfont
\char198\char229\char235\char224\char229\char236\
\char226\char224\char236\
\char243\char228\char224\char247\char232\char33
\end{document}

```

Lo que fue válido ayer será
mañana obsoleto.

Желаем вам удачи!

el cual es un archivo \LaTeX que utiliza conocimientos de codificación de caracteres, ya que es independiente de él, y conocimientos sobre codificación de fuentes, dado que los caracteres fueron reemplazados por secuencias del comando `\char` el cual invoca un carácter de una colección cuya posición coincida con la que recibe en su argumento. Esto puede verse como una herramienta de ofuscación, privacidad, seguridad o integridad de la información durante la transferencia de archivos de texto plano. En este caso nos sirve para despedir esta breve introducción con la frase en ruso cuya traducción es: **Le deseamos buena suerte!**

Capítulo 1

Cosas que debe saber

Abordaremos partiendo de Java, porque este libro es un complemento al libro *Programación en Java* del mismo autor, al igual que el capítulo *Expresiones regulares* del mencionado libro, pero a diferencia de este, el tema Unicode es más extenso y por eso se decidió separarlo, además se me hace más fácil explicar por este modo. Si usted no programa en Java, no se preocupe, la sintaxis del lenguaje son tangentes de la historia que pretendo contarle. Pues bien, empecemos.

1.1 Lo básico

En Java, los caracteres están a cargo de la clase `Character` (`java.lang`) cuyo primitivo es `char`, el acepta asignaciones de *literal de carácter* y *literal numérico entero*. Un literal de carácter se expresa como un carácter o una secuencia de escape, encerrado entre comillas simples.

```
char w = 'Ã';           // Literal de carácter único
char x = '\u00C5';       // Literal de carácter, escape Unicode (UTF-16)
char y = '\305';         // Literal de carácter, escape Octal (\u0000 a \u00ff)
char z = 197;            // Literal numérico entero
```

Como pudo ver, en Java el identificador de un carácter es el mismo pero expresado en diferentes sistemas numéricos, octal, decimal y hexadecimal. Además, nótese que dos de ellos son representadas mediante secuencias de escape (`'\305'` y `'\u00C5'`); No obstante también pueden ser asignados como literales numéricos cuya escritura es `0305` (octal) y `0x00C5` (hexadecimal). La diferencia solo tiene lugar en el octal, de ingresarse como secuencia de escape soporta el rango entre 0 a 255.

Un carácter Unicode a su vez cuenta con varios atributos, por mencionar algunos: *glifo*, *nombre*, *identificador*, *familia* y *categoría*.

El *glifo* es la representación gráfica final (en pantalla o impresa en un papel). El estándar de Unicode especifica una figura de referencia del cómo debe dibujar el carácter (si es que es dibujable).

```
char x = '\u221E';
System.out.printf("glifo: %s\n", x);
System.out.printf("name: %s\n", Character.getName(x));
```

glifo: ∞ name: INFINITY

Es de referencia ya que la forma final queda a disposición de una implementación en particular, por ejemplo: La tipografía, el cual define en base la fundición (Computer Modern,

Sans Serif, Times New Roman), tamaño (13, 14, 15 pt) o estilo (negrita, itálica); No satisfecho con la tipografía, puede usted mismo definir su propia implementación, esto lo veremos detalladamente en el capítulo 4.

El *nombre* es el común por el cual es más conocido o la que mejor la describen sus predecesoras (ASCII, Latin-1, Windows-1252, ISO 6429, etc.).

El *identificador*, conocido como *punto de código*, número que lo identifica a nivel del estándar en específico. Por ejemplo, el carácter INFINITY en la tabla IBM 437 está en la posición 236, en la tabla ASCII no existe (porque esta tabla solo tiene 128 caracteres) y para el estándar Unicode tiene la posición U+221E.

```
char x = '\u221E';
System.out.printf("codePoint: %d\n", (int) x);
System.out.printf("codePoint: U+ %04X\n", (int) x);
```

```
codePoint: 8734
codePoint: U+221E
```

En este punto es necesario aclarar a los blogueros, YouTuber's y demás personas que desconocen estos temas; nombrar "ASCII extendido" a una tabla que tiene 256 caracteres es un **error**. La tabla con nombre "ASCII extendido" **¡no existe!** Lo que *sí* existe son tablas, que, basándose en la ASCII original, de 128 caracteres (posiciones 0 al 127, 7 bits con 1 bit libre), definen sus propios 128 caracteres (posiciones 128 al 255, implementación del bit libre que dejó ASCII) resultando los 256 caracteres antes mencionados. Siendo un poco más técnicos, 11 11 11 11 es un número binario de 8 bits, es decir, 1 byte y para obtener en base 10 realizamos la operación $1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 256$. Estas tablas hijas (porque su madre es la tabla ASCII) tienen nombres muy específicos definidos por organismos reconocidos como la ISO, IBM, Microsoft, etc. Las más conocidas en Latinoamérica: Latin-1 (ISO/IEC 8858-1), Windows-1252 e IBM 850, esta última porque Windows 10 (versión 1903) para México la tiene como default cuya evidencia al presionar las teclas alt+213 le imprimirá ı (U+0131), de no ser así revítese las tablas de este libro para ver con cuál de ellas coincide su sistema operativo.

Un carácter, al igual que una persona, pertenece a una *familia*, algunos la llaman bloques, tablas, plantillas o Charts (para el idioma inglés).

```
char x = '\u221E';
System.out.printf("block: %s\n", Character.UnicodeBlock.of(x));

/* Lo mismo con la librería de mdt496 (bo.mdt496.unicode)
 * pero esta nos muestra toda la información de un bloque.
 * Además, permite consultar la URL donde reside el PDF del bloque.
 */
System.out.printf("block(mdt496-info): %s\n", MdtUnicodeBlock.of(x).info());
System.out.printf("block(mdt496-url): %s\n", MdtUnicodeBlock.of(x).pdfURL());
```

```
block: MATHEMATICAL_OPERATORS
block(mdt496-info): start=0x2200, range=2200..22FF, name=Mathematical Operators
block(mdt496-url): http://www.unicode.org/charts/U2200.pdf
```

La *categoría* es, al propósito que responde el carácter. Por ejemplo, hay algunos que son de control (Cc) otras son letras (Lo, Ll, Lu), números (No, Nd, Nl), símbolos matemáticos (Sm), etc.

```
char x = '\u221E';

/* El método getType, aunque útil, lastimosamente solo regresa el id */
System.out.printf("Type: %s\n", Character.getType(x));

/* Pero la librería de mdt496 (bo.mdt496.unicode)
```



```

* si nos regresa el nombre, e incluso toda la información.
*/
System.out.printf("Type(mdt496): %s\n", MdtUnicodeType.of(x));
System.out.printf("Type(mdt496-info): %s\n", MdtUnicodeType.of(x).info());

```

```

Type: 25
Type(mdt496): Math Symbol
Type(mdt496-info): id=25, name=Math Symbol, nameShort=Sm

```

1.2 Importancia

Hay contextos en los cuales es crítico saber cómo funciona la codificación de caracteres. Por ejemplo, si está empleando paquetes ofimáticos tipo WYSIWYG como Microsoft Word u OpenOffice, es el software quien lidia con la codificación de caracteres. En cambio en un entorno de lenguaje de marcado como \LaTeX , HTML y el mundo de la programación es el autor quien tiene que proporcionar la información respecto a la codificación de caracteres. Por ejemplo, vea el siguiente código HTML y explique porque la salida (Figura 1.1) imprime caracteres extraños.

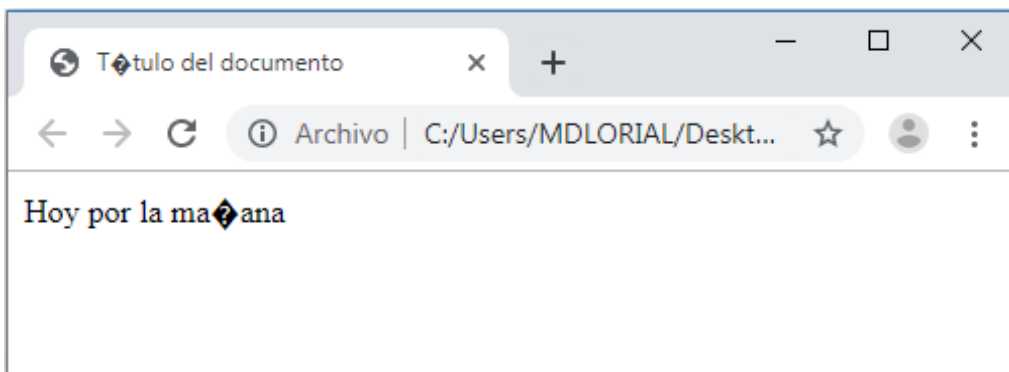
Ejemplo 1.1. Declarando el charset

```

<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <title>Titulo del documento</title>
  </head>
  <body>
    Hoy por la mañana
  </body>
</html>

```

Figura 1.1. Efectos de una declaración charset que no corresponde



Según el software editor (Figura 1.2) el archivo esta codificado en ANSI (azul), pero el programador está declarando en los metadatos que es utf-8 (rectángulo-rojo).

DOCUMENTO DEMOSTRATIVO
Adquiera la versión completa

Referencias

- [1] ISO/IEC 10646-1, *Information technology–Universal Multiple-Octet Coded Character Set (UCS)–Part 1: Architecture and Basic Multilingual Plane*, 2.^a ed., ISO Geneva, 2000.
- [2] ISO/IEC 10646-2, *Information technology–Universal Multiple-Octet Coded Character Set (UCS)–Part 2: Supplementary Planes*, ISO Geneva, 2001.
- [3] Unicode Consortium, *The Unicode Standard Version 12.0*, 2019, ISBN: 978-1-936213-22-1. dirección: <https://www.unicode.org/versions/Unicode12.0.0/UnicodeStandard-12.0.pdf> (visitado 15-06-2019).
- [4] J. Gosling, B. Joy, G. Steele, G. Bracha, A. Buckley, D. Smith y G. Bierman, *The Java™ Language Specification, Java SE 15 Edition*, Oracle America, Inc., 10 de ago. de 2020. dirección: <https://docs.oracle.com/javase/specs/jls/se15/jls15.pdf> (visitado 05-10-2020).
- [5] A. Jeffrey y F. Mittelbach, *inputenc.sty*, v1.2c, 2015.
- [6] T. Oetiker, H. Partl, I. Hyna y E. Schlegl, *Introducción no tan corta a L^AT_EX*. 2014, Versión: 5.03.
- [7] V. Volovich, W. Lemberg y L^AT_EX3 Project Team, *Cyrillic languages support in L^AT_EX*, 1999. dirección: <http://linorg.usp.br/CTAN/macros/latex/base/cyrguide.pdf> (visitado 09-12-2020).

Todos los documentos aquí expuestos se muestran con fines demostrativos y de marketing.
El autor asume toda responsabilidad por su contenido, eximiéndose a MDT496 ESTUDIOS y EBD SYSTEM.

Contacto

MDT496 ESTUDIOS | Marco Mendieta Parihuancollo

www.mdt496.wix.com/live

mdt496@gmail.com

www.youtube.com/mdt496

